UNIVERSIDADE FEDERAL DA FRONTEIRA SUL
CAMPUS DE CHAPECÓ
CURSO DE CIÊNCIA DA COMPUTAÇÃO

MATEUS DE LIMA

TÓPICOS ESPECIAIS EM COMPUTAÇÃO XIII

CHAT MQTT

CHAPECÓ
2025

Resumo

Este trabalho apresenta o desenvolvimento de uma aplicação de chat baseada no protocolo MQTT, com o objetivo de compreender e aplicar conceitos de comunicação assíncrona em sistemas distribuídos. O sistema foi implementado em Go e utiliza o broker Mosquitto como intermediário na troca de mensagens entre clientes. Cada usuário da aplicação é capaz de estabelecer conversas privadas ou participar de grupos, sendo a comunicação mediada por tópicos MQTT definidos para cada contexto. Além disso, foram implementados mecanismos para gerenciamento de presença (online/offline), controle de solicitações de chat e administração de grupos. Este trabalho demonstra como conceitos teóricos relacionados ao protocolo MQTT, amplamente utilizado em sistemas de Internet das Coisas (IoT), podem ser aplicados em um cenário prático, ressaltando sua importância e versatilidade.

INTRODUÇÃO

A aplicação foi desenvolvida na linguagem `Go`, utilizando o protocolo `MQTT` para troca de mensagens entre os usuários. O sistema implementa o modelo *publish/subscribe*, no qual os clientes publicam mensagens em tópicos e se inscrevem em tópicos para receber mensagens em tempo real. A aplicação gerencia a presença dos usuários, solicitações de chat, conversas privadas e comunicação em grupo. O uso do MQTT garante uma comunicação organizada, leve e escalável, fundamental para interações em tempo real.

ARQUITETURA DO SISTEMA E TÓPICOS DE CONTROLE

A arquitetura da aplicação foi organizada em quatro telas principais, cada uma com funções específicas para gerenciamento de usuários, grupos e conversas. A primeira tela apresenta a lista de usuários atualmente online, permitindo que o usuário selecione outro participante e envie solicitações para iniciar novas conversas privadas. A segunda tela exibe todas as solicitações de chats privados recebidas, oferecendo opções para aceitar ou recusar cada pedido, garantindo o controle sobre quem pode se comunicar diretamente. A terceira tela é dedicada ao gerenciamento de grupos, possibilitando a criação de novos grupos, a visualização de grupos existentes, o envio de solicitações para ingressar em grupos e a aprovação ou recusa de solicitações de entrada nos grupos que o usuário é proprietário. Por fim, a quarta tela representa a interface de chat em si, onde ocorrem o envio e o recebimento das mensagens, tanto em conversas privadas quanto em grupos, proporcionando uma experiência de comunicação em tempo real totalmente integrada.

A aplicação de chat inicia-se com o uso de tópicos de controle, que coordenam a criação de novas conversas. Por exemplo, um usuário que deseja iniciar um chat publica uma solicitação em um tópico específico associado ao destinatário `control/another_user`. Quando o destinatário aceita, um tópico privado é gerado dinamicamente `user1_User2_timestamp`,

permitindo a comunicação direta entre os dois usuários. Essa abordagem separa mensagens de controle das mensagens de conteúdo, evitando conflitos e tráfego desnecessário, também gerando tópicos únicos, onde a conversa pode fluir com relativa privacidade.

Além do tópico de controle dedicado à negociação de chats privados, o sistema utiliza outros dois tópicos centrais: `control/users` e `control/groups`. No primeiro, cada usuário conectado publica periodicamente seu nome e status de atividade, garantindo que todos os participantes da rede saibam quais usuários estão online e disponíveis para iniciar novas conversas. O tópico `control/groups` mantém informações sobre todos os grupos existentes, incluindo o proprietário (owner) e os membros. Os usuários podem solicitar entrada em um grupo publicando no tópico de controle do proprietário, que, ao aprovar, responde informando o tópico privado do grupo, onde as mensagens passam a ser trocadas.

A aplicação utiliza callbacks para gerenciar a recepção de mensagens em tempo real nos diferentes tópicos MQTT. Existem quatro callbacks: uma ouve o tópico `control/groups`, que mantém atualizadas as informações sobre os grupos existentes na rede; outra ouve o tópico `control/users`, que registra os usuários online e suas alterações de status; Sendo que as outras duas, uma ouve o tópico de controle individual do usuário `control/myUser`, que recebe mensagens relacionadas a solicitações de novos chats privados ou pedidos de entrada em grupos; e, por fim, a callback que ouve as mensagens de usuário, onde ocorrem as conversas propriamente ditas, tanto privadas quanto em grupos. Cada callback é responsável por processar as mensagens recebidas de seu respectivo tópico, atualizando as estruturas de dados da aplicação e garantindo que a interface do usuário reflita corretamente o estado atual das conversas e da rede.

Para garantir a consistência e continuidade da experiência do usuário, a aplicação implementa persistência local dos dados. Quando o usuário decide sair da aplicação digitando "0", uma última mensagem é publicada indicando que ele está offline, e a conexão com o broker MQTT é encerrada de forma segura. Em seguida, o estado completo da aplicação — incluindo chats, grupos, mensagens, solicitações e informações de usuários — é salvo em um arquivo local. Dessa forma, na próxima vez que o chat for iniciado, todos os dados são carregados, permitindo que o usuário retome suas conversas e mantenha o histórico de grupos e chats privados, garantindo uma experiência contínua e integrada.

FORMATO DAS MENSAGENS

O sistema adota um formato padronizado de mensagens baseado em **JSON**, garantindo compatibilidade e simplicidade na troca de informações entre clientes. Antes do envio, cada estrutura de mensagem é convertida (parseada) para JSON, e ao ser recebida pelo cliente destino, é novamente convertida de JSON para a respectiva estrutura interna. Essa abordagem elimina a necessidade de definir um protocolo próprio para o formato das mensagens, aproveitando a flexibilidade e legibilidade do JSON para trafegar dados de forma estruturada, leve e independente da linguagem de programação.

Foram definidos dois tipos principais de mensagens: **mensagens de sistema** e **mensagens de usuário**. As mensagens de sistema, representadas pela estrutura `SysMessage`, são responsáveis pela comunicação de controle dentro da aplicação, como solicitações para criação de novos chats, notificações de aceitação de conversas e envio dos tópicos privados nos quais as interações ocorrerão. Já as mensagens de usuário, modeladas pela estrutura `UserMessage`, transportam o conteúdo real das conversas — textos trocados entre participantes, tanto em chats privados quanto em grupos. Essa separação entre mensagens de controle e mensagens de conteúdo garante maior organização no tráfego de dados, evitando ambiguidades e facilitando o processamento e a atualização da interface do usuário.

O campo `Action`, presente na estrutura `SysMessage`, desempenha papel fundamental na identificação e tratamento das mensagens de sistema. Ele define o tipo de ação a ser executada, permitindo que o cliente interprete corretamente o propósito de cada mensagem recebida. Entre os valores possíveis para esse campo estão: `"GROUP_JOIN_ACCEPTED"`, `"REQUEST_JOIN_GROUP"`, `"PRIVATE_CHAT_ACCEPT"` e `"REQUEST_PRIVATE_CHAT"`. Por meio desse mecanismo, o sistema é capaz de distinguir, de forma simples e eficiente, mensagens relacionadas à administração de grupos, solicitações de entrada, ou estabelecimento de conversas privadas, garantindo assim um fluxo de comunicação consistente e bem definido.

Listing 1 – Estruturas de mensagens utilizadas na aplicação

```
type UserMessage struct {
    Read      bool
    From      string
    To        string
    Text      string
    IsGroup   bool
    GroupName string
}

type SysMessage struct {
    Action       string
    From         string
    PrivateTopic string
    Online       bool
    GroupName    string
}
```

CONCLUSÃO

A aplicação demonstra a viabilidade do protocolo MQTT para comunicação em tempo real, organizando o tráfego de mensagens por tópicos privados e de controle. O uso de tó-

picos específicos, combinados com a modelagem estruturada dos dados, permite gerenciar de forma eficiente a presença de usuários, solicitações de chat, conversas privadas e grupos. Essa abordagem evidencia a aplicação prática dos conceitos de mensageria assíncrona e *publish/subscribe* em sistemas distribuídos, garantindo escalabilidade e organização das interações na rede. Provando-se também um protocolo muito versátil, podendo ser usado em diferentes contextos.